Backups Using Storage Clusters

Joshua T. A. Davies
Michigan Technological University

Garrett W. Ransom
New Mexico Tech

Nicole M. Shaw
The University of New Mexico

Mentors: David Kennel, Cindy Valdez, Sonny Rosemond, Timothy Hemphill (DCS-CSD)

Introduction

Los Alamos National Laboratory generates huge amounts of data which represent immense time investments and are of vital importance to various defense and energy projects. To prevent any data loss, it is crucial for the laboratory to possess easily accessible backups. The current backup solutions are failing to keep pace with growing amounts of data, either taking too long to restore from or being too costly. Our project sought to implement a new design of commodity cluster storage system, using the open source projects ownCloud and GlusterFS, which is both cost and time efficient.

Design

Our system consisted of two storage tiers, the first acting as the primary storage space for backups and the second providing redundant duplicates of the first. The second tier was divided into two groups, which alternate on the task of synchronizing with the first tier. By keeping at least one of second tier's subdivisions offline and powered down at all times, we hoped to ensure that its copy of user data survives even disastrous failures.

ownCloud

- This open source application is designed to provide synchronization between various client devices and a cloud server.
- ownCloud client applications are provided for a variety of operating systems and will automatically synchronize user specified files and directories with the datastore of a remote server.
- The server provides a PHP web interface, through which files may be manipulated and restored in the event of deletion.

GlusterFS

- Open-source distributed file system, currently on version 3.5.1.
- Gluster is primarily managed by a system daemon, glusterd, which controls connections between nodes arranged into pools. Each node contributes one or more disk locations, referred to as bricks, to Gluster's logical volumes. One or more volume can be created per pool.
- For our system, ten of the nodes hosted Gluster volumes and were divided into three groups: the first (tier 1) containing two nodes and acting as the primary storage space of the system and the other two (collectively, tier 2) producting and storing copies of tier 1's data through Gluster's geo-replication process.

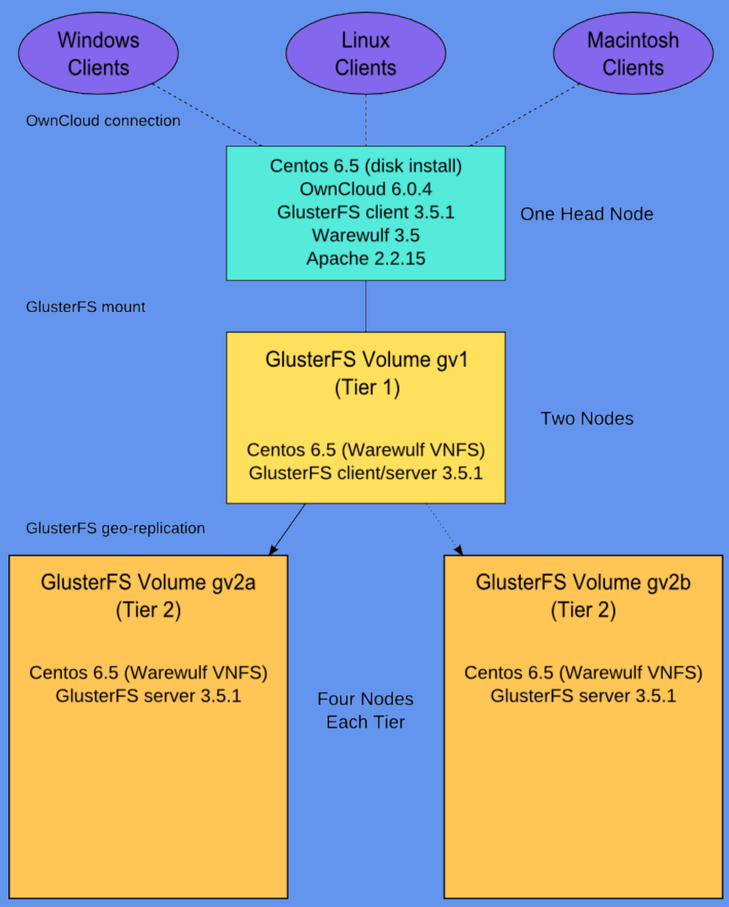

Figure 1: System Diagram

Scripts

We created four separate scripts to automate most system management:

- Tier Script
  - Acts on each specific Gluster tier.
  - Activates and deactivates Gluster volumes and geo-replication between tiers.
- Nodectl Script
  - Directly controls individual nodes within tiers.
  - Provides power state information, and remotely controls power up and down
- Switch Script
  - Uses tier and nodectl to alternate between tiers 2a and 2b. By integrating this script into a crontab, the process of tier switching was completely automated.
- Restore Script
  - Run manually, this script duplicates data from a tier 2 subdivision back to tier 1.

Issues

We encountered a number of setbacks during the installation and configuration of this design. Some of these problems have been overcome, while others were unable to be resolved and remain as hindrances for this design.

1. Originally, ownCloud failed to properly upload files larger than 2 GB in the web interface and larger than 4 GB in the desktop clients. An update to PHP resolved this issue for the web interface. Additionally, we discovered that large uploads were indeed possible via desktop clients, with our supposed issue actually being the result of a faulty client progress bar.

2. File permissions were not preserved from the original files in the local folder from all clients. ownCloud uses a global mask that will set all the permissions to default. We were not able to find a way to change this in the configurations.

3. We discovered that if we deleted a file in our local folder for the client, and then tried to restore the same file from the app client, files 2 GB and larger were corrupted both by the Mac OS X client , the Linux client, and the Windows client. Examination of hex dumps from uploaded and downloaded files revealed that corruption consistently began at byte 0x7fffffff, which sits precisely on the 2GB limit. Our examination of the corrupted areas in files has led us to believe that this error is the result of a buffer error, where portions of the file are continuously reread and used to construct the remaining data of of the file. Correcting this bug would require alteration of the ownCloud clients.

Conclusions

Our prototype system was successful in its basic tasks: to provide OwnCloud to various client operating systems, store data from ownCloud to GlusterFS volumes, and geo-replicate these volumes across tiers.. Additionally, Tier 2 was successfully automated to replicate data form Tier 1 as well as to conduct power cycling of its subdivisions. While the design shows promise, we believe that the issues encountered with ownCloud file permission preservation and data corruption render it impractical at present.

Future Work

- Submit a bug report to ownCloud to fix the file permissions and corruption issues.
- Further investigate the scaling capability of ownCloud and GlusterFS.
- Test the use of multiple ownCloud servers to handle large numbers of clients concurrently, making sure all servers can access the same data.
- Test Infiniband interconnects with Gluster, particularly whether geo-replication supports Infiniband

Contact Information
Joshua T. A. Davies: josuatadavies@gmail.com
Garrett W. Ransom: garrettwransom@gmail.com
Nicole M. Shaw: nmsaavedra26@gmail.com

LA-UR-14-25971